Dice Forge
Dossier technique

Team B « Otake »

Maxime CASTELLANO
Jason HAENLIN
Ruben HOURI
Vincent UNG

Synthèse du projet :

Nous avons pour cette livraison finale une version réduite du jeu Dice Forge avec actuellement la possibilité de faire jouer entre deux et quatre IA, qui sont capables de modifier leurs deux dés avec l'ensemble des faces de dé du jeu de base. Ensuite, l'ensemble des exploits sans token (exploits sans effet, à effet immédiat, le coffre et le marteau) sont utilisés et les bots sont capables de jouer une deuxième fois en dépensant deux pierres solaires.

Nous avons créé plusieurs bots qui se sont adaptés aux ajouts des nouvelles fonctionnalités du jeu. Le dernier en date est le bot Asta, qui a une stratégie pour jouer le marteau mais il n'utilise pas encore assez bien son or pour gagner. Cependant notre meilleur bot est Raichu, le bot de notre quatrième release, qui utilise un scénario précis pour pouvoir forger et jouer les exploits les plus chers qu'il peut acheter lors de son tour. Il y a aussi un bot aléatoire, Totoro pour voir l'intérêt des stratégies.

Nous avons fréquemment réalisé des tests unitaires (avec un coverage d'environ 80%) ce qui a permis de nous assurer une certaine fiabilité dans les nouvelles fonctionnalités ajoutées. Nous avons aussi utilisé des fichiers de configurations ce qui a facilité grandement la réalisation des tests notamment pour l'inventaire. Avec les derniers bots, nous avons réussi à implémenter la notion d'objet ce qui nous a permis de tester des petits morceaux de stratégie.

Nous sommes satisfaits de la structure globale du projet. En effet, nous nous sommes focalisés sur la structure pour faciliter l'ajout de nouvelles fonctionnalités. Nous avons aussi utilisé une nouvelle structure pour implémenter des stratégies plus facilement. Enfin, nous sommes satisfaits de l'encapsulation et de la responsabilité de chaque classe.

Nous n'avons cependant pas exploité toutes les fonctionnalités des stratégies et des fichiers de configuration pour faire des tests et des stratégies plus développées.

Pourquoi notre projet est de qualité ?

Dans ce projet, il nous était demandé d'implémenter un jeu de plateau et des IA non naïves jouant à ce jeu. Le point important était que les fonctionnalités implémentées du jeu soient utilisées intelligemment par le bot. En ce sens, toutes les fonctionnalités implémentées sont utilisées par au moins un des bots. Le meilleur bot pouvant utiliser toutes les fonctionnalités codées.

De plus, nous pensons que la partie la plus complexe de ce projet qui était la programmation des intelligences artificielles est une des parties les mieux structurées de notre projet. En effet, nous avons attaché de l'importance à l'unicité des fonctionnalités des classes et pensé à leur extensibilité. En particulier, pour les stratégies des bots, nous avons implémenté les stratégies comme des blocs de stratégie, et chaque stratégie correspond à un objet. Par exemple un bot aura un objet pour la façon dont il va forger et un objet pour la façon dont il va acheter les exploits. Il a aussi un objet qui correspond à son comportement par rapport à l'avancée du jeu appelé *Context*. Il utilisera des blocs de stratégie différents selon le tour dans lequel il est. Cette structure nous permet de pouvoir utiliser des comportements similaires pour des bots différents, et donc de voir si une stratégie est meilleure qu'une autre plus rapidement.

Nous avons aussi porté de l'importance à la façon dont le jeu était initialisé. Un package *util* à l'intérieur duquel une classe Config a été créée dans le but de pouvoir initialiser la partie comme nous le souhaitions avec l'utilisation d'un fichier JSON. Ainsi, nos tests sont plus faciles à mettre en place car nous pouvons choisir quels objets vont être utilisés dans le jeu. On peut alors initialiser les parties et les tests avec une configuration différente et donc maitriser le déroulement des tests malgré que les lancers de dés soient aléatoires lors de l'exécution du programme.

Dans le même package *util*, nous avons une classe *Guard* qui contrôle que les règles du jeu soient bien respectées, un système de token permet de donner la main au joueur, un joueur est donc contraint par ses niveaux d'accès aux classes pour ne pas faire des manœuvres non autorisées avec les objets du jeu et aussi par un objet *Guard* qui surveille s'il est bien autorisé à faire les actions à ce moment du jeu.

En quoi notre projet est mauvais ?

Le point faible de notre projet réside dans l'ajout de fonctionnalité. A l'heure actuelle, notre version du projet ne répond pas aux attentes de la 1er version du jeu incluant, la chasse, les cartes spéciales, les cartes avec des jetons et faces spéciales. Globalement nos bots malgré le travail réalisé sur la structure pour faciliter leur développement, auraient pu être plus intelligent.

Nous pouvons aussi dénoter les retards en termes de livraisons. Nous avons quelques fois été pris au dépourvu par la difficulté de la tâche. Cela nous a freiné considérablement dans notre développement et nous a parfois amené à changer notre structure global à cause du manque de recul. Si bien qu'à certain moment, nous avons dû repenser la structure pour l'adapter aux nouvelles fonctionnalités comme par exemple ; lors de l'ajout de nouvelles cartes demandant l'intervention du joueur pour choisir le dé ou pour choisir des ressources. Cette notion de callback (rappel) avait mal été gérée et a dû être repensée avec cette fois-ci, la prise d'un plus grand recul. Ces incidents impliquent un retard non négociable dans notre développement afin de produire un programme complet et de bonne qualité.

Rétrospective

L'utilisation de GitHub est une pratique que nous utilisons et continuerons à utiliser dorénavant car il s'agit d'un outil très pratique et devient vite indispensable dans un projet où le développement se fait en parallèle. Surtout lorsque nous utilisons les fonctionnalités avancées comme par exemple la mise en place d'Issues et de Milestones. De plus le tableau de bord automatique produit par l'automatique kanban permet de suivre l'avancement de notre projet de manière automatique en utilisant la liaison des *commits* aux issues. À cet outil s'ajoute la Javadoc que nous avons utilisé mais pas encore partout. Il faudrait l'utiliser constamment.

Cependant, certaines pratiques doivent être améliorées. La création des issues n'est pas toujours faite en amont et donc l'utilité d'un tel outil se voit réduite. Il nous a aussi été difficile de suivre un rythme stable dans notre avancée. Il faudrait que nous étudions plus profondément le sujet avant de mettre en place un quelconque emploie du temps. Cela nous a ralenti dans la progression du développement de nos IA.

Finalement, quelques habitudes doivent être supprimées, nous devons pour certains faire plus attention à nos *push* il nous est arrivé d'avoir du code qui ne passait pas les tests car seul l'exécution du Main avait été lancée avant le partage du code. Il faudrait aussi éviter d'envoyer du code complexe sans commentaire ou sans Javadoc. Cela augmente fortement le temps nécessaire pour la compréhension du code ; surtout quand les collaborateurs ne sont pas en train de travailler sur la partie même partie du code.